案例 16-二维图形几何变换算法

文档编写：霍波魏

校稿/修订：孔令德

时间 2019~2020

联系方式：QQ997796978

说明：本套案例由孔令德开发，原版本为 Visual C++6.0，配套于孔令德的著作《计算机图形学-基于 MFC 三维图形开发》一书。孔令德计算机工程研究所的学生霍波魏在学习计算机图形学期间，对本套案例进行了升级并编写了学习文档。现在程序的编写和程序的解释都是基于 Windows 10 操作系统，使用 Microsoft visual studio 2017 平台的 MFC（英文版）开发。

一、知识点

1. 矩阵相乘

在进行图形几何变换时需要进行矩阵相乘运算。若 A 为$m \times n$矩阵，B 为$n \times p$矩阵，则他们的乘积AB(有时记做$A \cdot B$）会是一个$m \times p$矩阵。其乘积矩阵的元素如下面式子得出：

$$(AB)_{ij} = \sum_{r=1}^{n} a_{ir} b_{rj} = a_{i1} b_{1j} + a_{i2} b_{3j} + \cdots + a_{in} b_{nj}$$

2.平移变换矩阵

平移变换是指将$P(x,y)$点移动到$P'(x',y')$位置的过程，平移变换的坐标表示为：$\begin{cases} x' = x + T_x \\ y' = y + T_y \end{cases}$

3.比例变换矩阵

比例变换是指$P(x,y)$相对于坐标元带你 O，沿 x 方向缩放 S_x 倍，沿 y 方向缩放 S_y 倍，得到$P'(x',y')$点的过程

坐标表示如下：$\begin{cases} x' = x \cdot S_x \\ y' = y \cdot S_y \end{cases}$

4.旋转变换矩阵

旋转变换是$P(x,y)$点相对于坐标原点 O 旋转一个角度 β (逆时针方向为正，顺时针方向为负)，得到$P'(x',y')$点的过程，对于$P(x,y)$点，极坐标表示为：$\begin{cases} x = r\cos\alpha \\ y = r\sin\alpha \end{cases}$

旋转变换的坐标表示为：$\begin{cases} x' = r\cos(\alpha + \beta) = x\cos\beta - y\sin\beta \\ y' = r\sin(\alpha + \beta) = x\sin\beta + y\cos\beta \end{cases}$

二、案例描述

本案例用使用二维图形几何变换算法，借助 Dialog 对话框，实现一个二维图形的几何变换（包括：平移变换、比例变换、旋转变换）。

三、实现步骤

1. 添加二维变换类 CTransform2，在 CTransform2 类中添加平移变换、比例变换、旋转变换函数。
2. 添加绘制直线函数 Cline 类。
3. 在资源视图中添加 Dialog 资源，在该资源中添加对话框，给对话框添加变量,添加函数功能。
4. 在 CTestView 中计算顶点坐标，进行边界检测，绘制图形。

四、主要算法

1.CTransform2 类

```
public:
    CTransform2();
    virtual ~CTransform2();
    void SetMat(CP2d* p, int n);
    void Identity();
    void Translate(double tx, double ty);//平移变换矩阵
    void Scale(double sx, double sy);//比例变换矩阵
    void Scale(double sx, double sy, CP2d p);//相对于任意点的比例变换矩阵
    void Rotate(double beta);//旋转变换矩阵
    void Rotate(double beta, CP2d p);//相对于任意点的旋转变换矩阵
    void Reflect0();//原点反射变换矩阵
    void ReflectX();//X轴反射变换矩阵
    void ReflectY();//Y轴反射变换矩阵
    void Shear(double b, double c);//错切变换矩阵
    void MultiMatrix();//矩阵相乘
public:
    double T[3][3];
    CP2d* POld;
    int num;
void CTransform2::SetMat(CP2d * p, int n)
{
    POld = p;
    num = n;
}
void CTransform2::Identity()//单位矩阵
```

```
{
    T[0][0] = 1.0; T[0][1] = 0.0; T[0][2] = 0.0;
    T[1][0] = 0.0; T[1][1] = 1.0; T[1][2] = 0.0;
    T[2][0] = 0.0; T[2][1] = 0.0; T[2][2] = 1.0;
}
void CTransform2::Translate(double tx, double ty)//平移变换矩阵
{
    Identity();
    T[2][0] = tx;
    T[2][1] = ty;
    MultiMatrix();
}
void CTransform2::Scale(double sx, double sy)//比例变化矩阵
{
    Identity();
    T[0][0] = sx;
    T[1][1] = sy;
    MultiMatrix();
}
void CTransform2::Scale(double sx, double sy, CP2d p)
//相对于任意点的整体比例变换矩阵
{
    Translate(-p.x, -p.y);
    Scale(sx, sy);
    Translate(p.x, p.y);
}
void CTransform2::Rotate(double beta)//旋转变换矩阵
{
    Identity();
    double rad = beta * PI / 180;
    T[0][0] = cos(rad);T[0][1] = sin(rad);
    T[1][0] = -sin(rad);T[1][1] = cos(rad);
    MultiMatrix();
}
void CTransform2::Rotate(double beta, CP2d p)//相对于任意点的旋转变换矩阵
{
    Translate(-p.x, -p.y);
    Rotate(beta);
    Translate(p.x, p.y);
}
void CTransform2::Reflect0()//原点反射变换矩阵
{
    Identity();
    T[0][0] = -1;
```

```
        T[1][1] = -1;
        MultiMatrix();
    }
    void CTransform2::ReflectX()//X轴反射变换矩阵
    {
        Identity();
        T[0][0] = 1;
        T[1][1] = -1;
        MultiMatrix();
    }
    void CTransform2::ReflectY()//Y轴反射变换矩阵
    {
        Identity();
        T[0][0] = -1;
        T[1][1] = 1;
        MultiMatrix();
    }
    void CTransform2::Shear(double b, double c)//错切变换还矩阵
    {
        Identity();
        T[0][0] = b;
        T[1][1] = c;
        MultiMatrix();
    }
    void CTransform2::MultiMatrix()//矩阵相乘
    {
        CP2d* PNew = new CP2d[num];
        for (int i = 0; i < num; i++)
        {
            PNew[i] = POld[i];
        }
        for (int j = 0; j < num; j++)
        {
            POld[j].x = PNew[j].x * T[0][0] + PNew[j].y * T[1][0] + PNew[j].w *
    T[2][0];
            POld[j].y = PNew[j].x * T[0][1] + PNew[j].y * T[1][1] + PNew[j].w *
    T[2][1];
            POld[j].w = PNew[j].x * T[0][2] + PNew[j].y * T[1][2] + PNew[j].w *
    T[2][2];
        }
        delete[]PNew;
  }
```

2.CTestView 类

```
public:
```

```
    void DrawObject(CDC* pDC);//绘制图形
    void DoubleBuffer();//双缓冲
    void ReadPoint();//计算顶点做表
    void BorderCheck();//边界检测
protected:
    int nClientWidth, nClientHeight;//屏幕客户区宽度和高度
    int nHWidth, nHHeight;//屏幕客户区的半宽和半高
    CP2d*  P;//变换点
    int    directionX, directionY;//位移方向
    double translateX, translateY, rotate;
    double scale;
    int    degree;
    double nRadius;//图形半径
    CTransform2 tran;//二维几何变换对象
void CTestView::DoubleBuffer() //双缓冲
{
    CDC* pDC = GetDC();
    CRect rect;//定义客户区
    GetClientRect(&rect);//获得客户区的大小
    GetClientRect(&rect);
    nClientWidth = rect.Width();//屏幕客户区宽度
    nClientHeight = rect.Height();//屏幕客户区高度
    nHWidth = nClientWidth / 2;//屏幕客户区半宽
    nHHeight = nClientHeight / 2;//屏幕客户区半高
    CDC memDC;
    memDC.CreateCompatibleDC(pDC);
    CBitmap NewBitmap, *pOldBitmap;
    NewBitmap.CreateCompatibleBitmap(pDC, nClientWidth, nClientHeight);
    pOldBitmap = memDC.SelectObject(&NewBitmap);
    ReadPoint();//计算图形顶点坐标
    //平移变换
    tran.Translate(translateX, translateY);
    //相对于任意点的旋转变换
    tran.Rotate(rotate, CP2d(translateX, translateY));
    //相对于任意点的比例变换
    tran.Scale(scale, scale, CP2d(translateX, translateY));
    DrawObject(&memDC);
    BorderCheck();
    pDC->BitBlt(0, 0, nClientWidth, nClientHeight, &memDC, 0, 0, SRCCOPY);
    memDC.SelectObject(pOldBitmap);
    NewBitmap.DeleteObject();
    if (P != NULL)
    {
        delete[]P;
```

```
            P = NULL;
        }
    }
    void CTestView::DrawObject(CDC* pDC) //绘制图形
    {
        CLine *line = new CLine;
        for (int i = 0; i <= degree - 2; i++)
        {
            for (int j = i + 1; j <= degree - 1; j++)
            {
                line->MoveTo(pDC, ROUND(nHWidth + P[i].x), ROUND(nHHeight - P[i].y),
            CRGB(1.0, 0.0, 1.0));
                line->LineTo(pDC, ROUND(nHWidth + P[j].x), ROUND(nHHeight - P[j].y),
            CRGB(1.0, 0.0, 1.0));
            }
        }
        delete line;
    }
    void CTestView::ReadPoint() //计算顶点做表
    {
        double Dtheta = 2 * PI / degree;
        P = new CP2d[degree + 1];
        for (int i = 0; i < degree; i++)
        {
            P[i].x = nRadius * cos(i*Dtheta);
            P[i].y = nRadius * sin(i*Dtheta);
        }
        P[degree].x = 0; P[degree].y = 0;//图形中心点
        tran.SetMat(P, degree + 1);
    }
    void CTestView::BorderCheck() //边界检测
    {
        double TempR = nRadius * scale;
        if (fabs(P[degree].x) + TempR > nHWidth)
        {
            directionX *= -1;
            translateX += fabs(fabs(P[degree].x) + TempR - nHWidth) * directionX;
                            //判断球体水平越界
        }
        if (fabs(P[degree].y) + TempR > nHHeight)
        {
            directionY *= -1;
            translateY += fabs(fabs(P[degree].y) + TempR - nHHeight) * directionY;
                            //判断球体垂直越界
```

```
    }
}
void CTestView::OnTimer(UINT_PTR nIDEvent)
{
    CTestDoc* pDoc = GetDocument();
    if (((CMainFrame*)AfxGetMainWnd())->IsPlay)
    {
        degree = pDoc->m_Degree;
        translateX += pDoc->m_TranslateX * directionX;
        translateY += pDoc->m_TranslateY * directionY;
        rotate += pDoc->m_Rotate;
        scale = pDoc->m_Scale;
        DoubleBuffer();
    }
    CView::OnTimer(nIDEvent);
}
BOOL CTestView::OnEraseBkgnd(CDC* pDC)//禁止背景刷新
{
    // TODO: 在此添加消息处理程序代码和/或调用默认值
    return  TRUE;
}
```

五、实现效果

二维图形几何变换算法效果如图 16-1 所示。

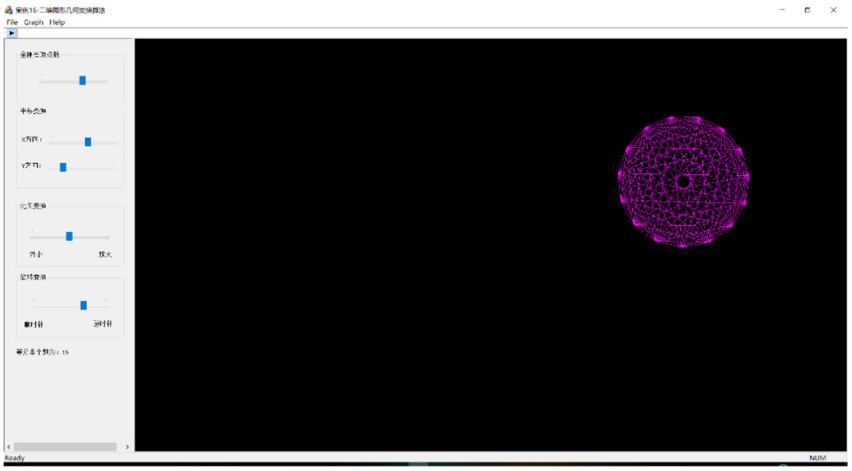

图 16-1 二维图形几何变换算法效果图

六、遇到的问题及解决方案

1. Dialog 对话框的创建以及功能函数的编写

在资源视图中创建 dialog 对话框，将其 Border 的默认值由“对话框外框”改为“None”，在对话框中添加 Group Box、Slider control 以及 Static Text，接

下来需要给对话框添加变量，然后对对应的变量写对应的功能，如图 16-2 所示。

```
CSliderCtrl m_degree;
CSliderCtrl m_translateX;
CSliderCtrl m_translateY;
CSliderCtrl m_scale;
CSliderCtrl m_rotate;
CStatic m_point;
```

图 16-2 为对话框添加的变量

```
void CLeftPortion::OnInitialUpdate()
{
    CFormView::OnInitialUpdate();
    // TODO: Add your specialized code here and/or call the base class
    //设置左窗格滑动条的范围及初始值
    m_degree.SetRange(1, 4, TRUE);
    m_degree.SetPos(3);

    m_translateX.SetRange(0, 10, TRUE);
    m_translateX.SetPos(6);
    m_translateX.SetTicFreq(2);
    m_translateX.SetPageSize(2);

    m_translateY.SetRange(0, 10, TRUE);
    m_translateY.SetPos(2);
    m_translateY.SetTicFreq(2);
    m_translateY.SetPageSize(2);

    m_scale.SetRange(-2, 2, TRUE);
    m_scale.SetPos(0);

    m_rotate.SetRange(-10, 10, TRUE);
    m_rotate.SetPos(4);
    m_rotate.SetTicFreq(4);
    m_rotate.SetPageSize(5);

    CString str("");
    str.Format(_T("等分点个数为：%d"), m_degree.GetPos() * 5);
    m_point.SetWindowText(str);
    UpdateData(FALSE);
}
```

```
void CLeftPortion::OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
{
    // TODO: 在此添加消息处理程序代码和/或调用默认值
    CTestDoc* pDoc = (CTestDoc*)CFormView::GetDocument();
    UpdateData();
    switch (m_degree.GetPos())
    {
    case 1:
        pDoc->m_Degree = 5;
        break;
    case 2:
        pDoc->m_Degree = 10;
        break;
    case 3:
        pDoc->m_Degree = 15;
        break;
    case 4:
        pDoc->m_Degree = 20;
        break;
    }
    pDoc->m_TranslateX = m_translateX.GetPos();
    pDoc->m_TranslateY = m_translateY.GetPos();
    switch (m_scale.GetPos())
    {
    case 2:
        pDoc->m_Scale = 1.4;
        break;
    case 1:
        pDoc->m_Scale = 1.2;
        break;
    case 0:
        pDoc->m_Scale = 1.0;
        break;
    case -1:
        pDoc->m_Scale = 0.8;
        break;
    case -2:
        pDoc->m_Scale = 0.6;
        break;
    }
    pDoc->m_Rotate = m_rotate.GetPos();
    CString str("");
    str.Format(_T("等分点个数为: %d"), pDoc->m_Degree);
    m_point.SetWindowText(str);
    UpdateData(FALSE);
    CFormView::OnHScroll(nSBCode, nPos, pScrollBar);
}
```

图 16-3 设置对话框功能

2. MFC 单文档框架三个类的功能(本是四个，APP 类没用到)

CMainFrame 是视图类即 View 类的父窗口，视图就显示在 CMainFrame 的客户区中；document/view 模式是为了在逻辑上让数据和显示分开；一般在 document 里，定义 document 类的成员变量，来存数据，并用 View 来显示；在 View 里，用 GetDocument 来获取与之对应 document 的指针，进而可以访问 document 的成员变量，从而进行显示 Document/View；一个视图类只能跟一个文档类相联系，而一个文档类可以跟多个视图类相联系；CMainframe 类和 CView 类都是从 CWnd 继承而来，都是窗口类。

七、案例心得

该案例是二维图形几何变换的第一个案例，该案例主要的类是 CTransform2 类，在该类中定义了矩阵相乘、平移变换矩阵、比例变换矩阵、旋转变换矩阵、反射变换矩阵和错切变换矩阵函数，但是在本案例中只用到了平移变换矩阵、比例变换矩阵和旋转变换矩阵，矩阵相乘则是其他变换矩阵的基础；在该案例中，第一次用到了对话框和对话框函数，也就用到了 MFC 的单文档框架；相比于之前的图形学程序，这次也更进一步的了解了、使用了 MFC 框架的一些基础类的功能。